Radial Basis Function Neural Networks: A Review

Gholam Ali Montazer[1], Davar Giveki[2], Maryam Karami[3], Homayon Rastegar[4]

[1]School of Engineering, Department of Information Technology Engineering, Tarbiat Modares University, Tehran, Iran

[2]University of Lorestan, Department of Computer Engineering, KhoramAbad, Lorestan, Iran

[3]Department of computer engineering, university of Afarinesh, Boroujerd, Iran

[4]Department of computer engineering, university of Afarinesh, Boroujerd, Iran

montazer@modares.ac.ir, davood.giveki@gmail.com, Mkarami916@gmail.com, rhomayon@gmail.com

Abstract

Radial Basis Function neural networks (RBFNNs) represent an attractive alternative to other neural network models. One reason is that they form a unifying link between function approximation, regularization, noisy interpolation, classification and density estimation. It is also the case that training RBF neural networks is faster than training multi-layer perceptron networks. RBFNN learning is usually split into an unsupervised part, where center and widths of the Gaussian basis functions are set, and a linear supervised part for weight computation. This paper reviews various learning methods for determining centers, widths, and synaptic weights of RBFNN. In addition, we will point to some applications of RBFNN in various fields. In the end, we name software that can be used for implementing RBFNNs.

Keywords: Radial Basis Function Neural Networks (Rbfnns), Learning Algorithm, Center of Gaussian Function, Width of The Gaussian Function.

1. Introduction

Artificial neural networks are an attempt to emulate the processing capabilities of biological neural systems. The basic idea is to realize systems capable of performing complex processing tasks by interconnecting a high number of very simple processing elements which might even work in parallel. They solve cumbersome and intractable problems by learning directly from data. An artificial neural network usually consists of a large amount of simple processing units, i.e., neurons, with mutual interconnections. It learns to solve problems by adequately adjusting the strength of the interconnections according to input data. Moreover, it can be easily adapted to new environments by learning. At the same time, it can deal with information that is noisy, inconsistent, vague, or probabilistic. These features motivate extensive research and developments in artificial neural networks.

The main features of artificial neural networks are their massive parallel processing architectures and the capabilities of learning from the presented inputs. They can be utilized to perform a specific task only by means of adequately adjusting the connection weights, i.e., by training them with the presented data. For each type of artificial neural network, there exists a corresponding learning algorithm by which we can train the network in an iterative updating manner. Those learning algorithms fit into two main categories: supervised learning and unsupervised learning.

For supervised learning, not only the input data but also the corresponding target answers are presented to the network. Learning is done by the direct comparison of the actual output of the network with known

[1] Name, address etc. of corresponding author:
Gholam Ali Montazer, Email: montazer@modares.ac.ir, , Tarbiyat Modares University of Tehran, Tehran, Iran.
Telephone: +982155749730
MobilePhone: +98 9123230540

correct answers. This is also referred to as learning with a teacher. In contrast, if only input data without the corresponding target answers are presented to the network for learning, we have unsupervised learning. In fact, the learning goal is not defined at all in terms of specific correct examples. The available information is in the correlations of the input data. The network is expected to create categories from these correlations and to produce output signals corresponding to the input category.

Neural networks have been successfully employed to solve a variety of problems. They are systems of interconnected, simple processing elements. A neural network aims to learn the nonlinear mapping from an input space describing the sensor information onto an output space describing the classes to which the inputs belong. In between, radial basis function neural networks are one of the most popular and high performance neural networks [1].

RBFNNs were first introduced by Powell [25] to solve the interpolation problem in a multi-dimensional space requiring as many centers as data points. Later Broom head and Lowe [1] removed the 'strict' restriction and used less centers than data samples, so allowing many practical RBFNNs applications in which the number of samples is very high. An important feature of RBFNNs is the existence of a fast, linear learning algorithm in a network capable of representing complex non-linear mapping. At the same time, it is also important to improve the generalization properties of RBFNNs [2,26]. Today RBFNNs have been a focus of study not only in numerical analysis but also in machine learning researchers. Being inherited from the concept of biological receptive field [27] and followed, Park and Sandberg prove, "RBFNNs were capable to build any non-linear mappings between stimulus and response" [28].

In the last decade, different RBF learning methods have been proposed to reduce the number of hidden neurons by means of smarter selection of neuron centers and widths. Determination of required number of hidden units, their centers and spreads are the main parts of an RBF learning rule. In this paper we briefly study various learning algorithm proposed for training radial basis function neural networks.

The rest of the paper is organized as following. In Section 2, radial basis function neural network is introduced. Section 3 discusses different radial basis function neural networks learning algorithms. Finally, the paper is concluded in Section 4.

2. RBF neural network architecture

The idea of RBFNNs is derived from the theory of function approximation. The Euclidean distance is computed from the point being evaluated to the center of each neuron, and a radial basis function (RBF) (also called a kernel function or Gaussian function) is applied to the distance to compute the weight (influence) for each neuron. The radial basis function is so named because the radius distance is the argument to the function. In other words, RBFs represent local receptors; its output depends on the distance of the input from a given stored vector. That means, if the distance from the input vector $\vec{x}$ to the center $\vec{c_i}$ of each RBF φ_j i.e., $\|\vec{x} - \vec{c_i}\|$ is equal to 0 then the contribution of this point is 1, whereas the contribution tends to 0 if the distance $\|\vec{x} - \vec{c_i}\|$ increases.

A radial basis function neural network is a three-layer network. As it is depicted in Fig. 1., the layers include: input layer, hidden layer and output layer (summation layer).

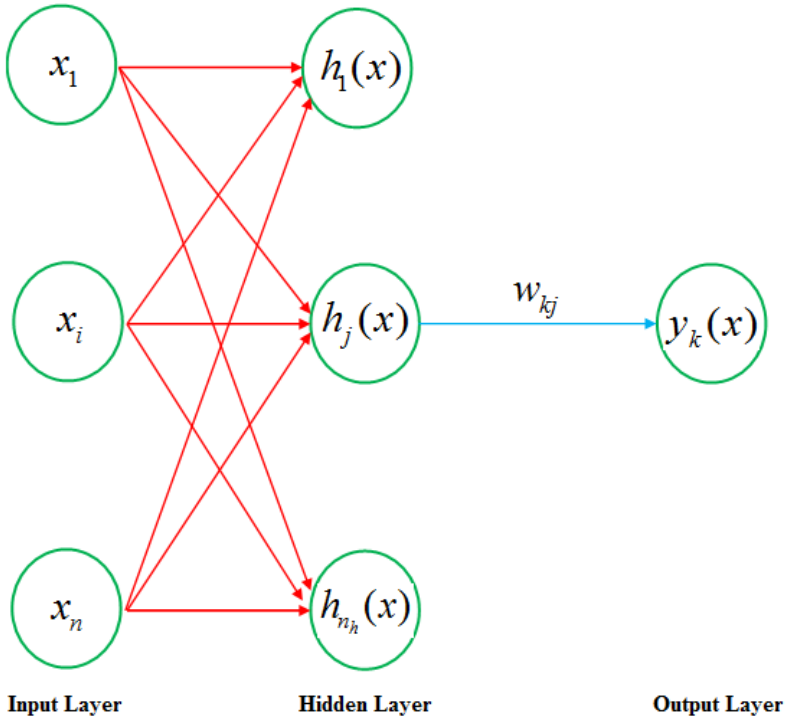

Figure 1. Architecture of a radial basis function network

The input layer can have more than one predictor variable where each variable is associated with one independent neuron. The function of the input layer is to propagate input vectors to the hidden layer. The hidden layer includes a number of radial basis function units (n_h) with Gaussian kernels and bias (b_k). The j-th Gaussian function is determined by a center (c_j) and a width (σ_j). The RBFNN classifier starts by comparing the Euclidean distance between the input vector (x) and the center of the Gaussian function (c_j) and performs the nonlinear transformation in the hidden layer as follows:

$$h_j(x) = \exp\left(-\|x - c_j\|^2 / \sigma_j^2\right), \quad (1)$$

where h_j is the notation for the output of the j-th neuron in the hidden layer? The linear operation of the output layer is given as follows:

$$y_k(x) = \sum_{j=1}^{n_h} w_{kj}.h_j(x) + b_k, \quad (2)$$

where $y_k$is the k-th output unit for the input vector x, w_{kj} is the weight connection between the k-th output unit and the j-th hidden layer unit, and b_k is the bias. So, in the output layer k-th neuron is associated with weights $(w_{k1}, w_{k2}, \ldots, w_{kn_h})$. The value coming out of a neuron in the hidden layers is multiplied by the weight associated with the neuron and passed to the summation which adds up the weighted values and presents this sum as the output of the network. A bias value is multiplied by a weight b_k and fed into the output layer.

3. RBFNN training algorithms

Learning or training a neural network is a process by means of which the network adapts itself to a stimulus by making proper parameter adjustment, resulting in the production of desired response. Hence, to get the simila approximation/classification accuracy and in

addition to the required number of RBF units, the following parameters are determined by the training process of RBFNNs [1]:

1. The number of neurons in the hidden layer. [Ideally the number of neurons (M) in the hidden layer should be much less than data points (N)]; The number of hidden neurons in each RBFNN is determined by conducting different experiments.

2. The coordinates of the center of each hidden node, determined by the training algorithm;

3. The width (spread) of each RBF in each dimension, determined by the training algorithm; and

4. The weights applied to the RBF outputs as they are passed to the output layer.

3.1. RBF Kernels

The Gaussian kernel is a usual choice for kernel functions. The commonly used RBFs are expressed in Table 1.

Kernel	Mathematical representation
Generalized multi-quadric function	$\Phi(r) = (r^2 + c^2)^{\beta}, \quad c > 0, 0 < \beta < 1$
Generalized inverse multi-quadric function	$\Phi(r) = (r^2 + c^2)^{-\alpha}, \ 0 < \alpha < c$
Thin plate spline basis function	$\Phi(r) = r^2 \ln(r)$
Cubic function	$\Phi(r) = r^3$
Linear function	$\Phi(r) = r$

In multi-quadric function the matrix representation of basis function has an important spectral property: it is almost negative definite. Franke [29] has found that this radial basis function provides the most accurate interpolation surface in two dimensions. Also, he found that the inverse multi-quadric basis function can provide excellent approximations, even when the number of centers is small. However, author presents that sometimes a large value of σ can be useful [30]. In contrast, there is no good choice of σ known at present in the case of multi-quadric basis function.

The thin plate spline basis function has more global nature than the Gaussian function i.e., a small perturbation of one of the control points always affect the coefficients corresponding to all other points as well. Similarly, the polynomial basis functions like cubic and linear has some degree of influence in certain applications.

An overview of RBFs and its corresponding models are described in [31]. It is observed that in most of the neural network literature that the Gaussian RBFs is widely used in diversified domain such as medical/biological science, control systems, engineering, etc.

3.2. Computing centers, widths and weights of a RBFNN

A main advantage of RBF networks is choosing suitable hidden unit/basis function parameters without having to perform a full non-linear optimization of the whole network. The coordinates of center of each hidden-layer in RBF function can be calculated by using any of the following unsupervised methods.

3.2.1 Fixed centers selected at random

This is a simple and fast approach for setting the RBF parameters, where the centers are kept fixed at M points selected at random from the N data points. Specifically, we can use normalized RBFs centered at $\{c_j\}$ defined by

$$\phi_j(x) = exp\left(-\frac{\|x-c_j\|^2}{2\sigma_j^2}\right), \quad (3)$$

where $\{c_j\} \subseteq \{X^P\}$ and σ_j is width of the Gaussian functions [3,6].

3.2.2 Clustering

Clustering techniques can be used to find a set of centers which more accurately reflect the distribution of the data points. The K-means clustering algorithm [32] selects the number *K* of centers in advance, and then follows a simple re-computation procedure to divide the data points $\{X^P\}$ into *K* disjoint subsets S_j and N_j data points in order to minimize the sum of squared clustering function.

$$J = \sum_{j=1}^{K} \sum_{p \in S_j} \|X^p - c_j\|^2, \quad (4)$$

where, c_j is the mean/centroid of the data points in set S_j given by the Equation (5):

$$c_j = \frac{1}{N} \sum_{p \in S_j} X^p, \quad (5)$$

There are, however, two intrinsic disadvantages associated with the use of K-means. The first is due to its iterative nature, which can lead to long convergence times, and the second originates from its inability to automatically determine the number of RBF centers, thus resulting in a time consuming trial-and-error procedure for establishing the size of the hidden layer.

A multitude of alternative techniques have been proposed to tackle these disadvantages. One way is to use some improved unsupervised methods [33] such as: fuzzy clustering [34-37], self organizing map (SOM) [38], particle swarm optimization (PSO)-based subtractive clustering [9-12], dynamic K-means clustering algorithm [39], improved K-means algorithm [33], K-harmonic clustering [40], and self-additive clustering [41], have been used for center selection in RBFNNs.

In [6], the Optimum Steepest Decent (OSD) method was introduced for calculating the connection weights in the hidden layer of the RBFNN classifier. This method uses an optimum learning rate in each epoch of the training process. The proposed method used RBF neural network to design a helicopter identification system. In spite of its fast learning process and high performance, it suffers from random selection of the centers and widths of the RBF units, which decreases the efficiency of the proposed RBFNN. As a follow-up work, authors in [7] proposed a three-phase learning algorithm to improve the performance of the OSD. This method uses *k*-means and *p*-nearest neighbor algorithms to determine the centers and the widths of RBF units, respectively, which results in a greater precision in initializing RBF unit centers and widths. This algorithm guarantees reaching the global minimum in the weight space, however, the sensitivity of *k*-means to the center initialization can lead the algorithm to get stuck in a local minimum which results in a suboptimal solution. As a result, Montazer et al. [8] proposed using fuzzy c-means clustering instead of k-means clustering. They believe that using fuzzy c-means the center of the Gaussian functions can more accurately be determined. Although the proposed method could achieve high performance, it was very slow and sensitive to center initialization of centers. Hence, it cannot be applicable to feature vectors of high dimensional. Therefore, Fathi and Montazer [9] investigated particle swarm optimization (PSO) technique for adjusting the centers of the Gaussian functions while *p*-nearest neighbor algorithm was used for width adjustment.

In [9], the velocity and position updating rules are given by:

$$\begin{cases} v_{id}^{k+1} = \omega v_{id}^{k} + c_1 r_1 (pbest_{id}^{k} - x_{id}^{k}) + c_2 r_2 (gbest_{d}^{k} - x_{id}^{k}) & (6) \\ x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1}, \quad i = 1, 2, ..., n & (7) \\ \omega = \omega_{\max} - \dfrac{k}{k_{\max}} (\omega_{\max} - \omega_{\min}) & (8) \end{cases}$$

where the current position of the particle i in the k-th iteration is x_{id}^{k} and v_{id}^{k} is the current velocity of the particle which is used to determine the new velocity v_{id}^{k+1}. The c_1 and c_2 are acceleration coefficients. The r_1 and r_2 are two independent random numbers uniformly distributed in the range of [0,1]. In addition, $v_i \in [-v_{\max}, v_{\max}]$, where $v_{\max}$ is a problem-dependent constant defined to clamp the excessive roaming of particles. The $pbest_{id}^{k}$ is the best previous position along the d-th dimension of the particle i in the iteration k (memorized by every particle); $gbest_{d}^{k}$ is the best previous position among all the particles along the d-th dimension in the iteration k (memorized in a common repository). The ω_{max} and ω_{min} are the maximum and the minimum of ω, respectively. The k_{max} is the maximum number of iterations.

In spite of its high optimization ability, PSO can get trapped in a local optimum which slows down the convergence speed. In addition, using p-nearest neighbor algorithm for computing the widths of RBF units, results in a loss of information about the spatial distribution of the training dataset; and as a consequence, the computed widths do not make a major contribution in the classification performance of very complicated data such as images. To alleviate this drawback, Montazer and Giveki [10] proposed a new adaptive version of particle swarm optimizer capable of working with high dimensional data. The obtained results show fast convergence speed, better and same network response in fewer train data which states the generalization power of the improved neural network proposed in [10].

The proposed adaptive strategy works as follows:

$$\begin{cases} v_{id}^{k+1} = \omega_i^{k} v_{id}^{k} + c_1 r_1 (pbest_{id}^{k} - x_{id}^{k}) + c_2 r_2 (gbest_{d}^{k} - x_{id}^{k}) & (9) \\ x_{id}^{k+1} = x_{id}^{k} + \mu_i^{k} v_{id}^{k+1}, \quad i = 1, 2, ..., n & (10) \end{cases}$$

The adaptive strategy is a method to dynamically adjust the inertia weight factor ω and the new velocity v_{id}^{k+1} by introducing the coefficient μ.

The inertia weight ω has a great influence on the optimal performance. Empirical studies of PSO with inertia weight have shown that a relatively large ω has more global search ability while a relatively small ω results in a faster convergence. Although in Equation (6), ω is adaptive, it is updated using the linear updating strategy of Equation (8). As a result, ω is just relevant to the current iteration and maximum number of iterations (k and k_{max}) and cannot adapt to the characteristics of complexity and high nonlinearity. If the problem is extremely complex, the global search ability is insufficient in the later iteration. Therefore, in order to overcome the above defects, an improved method for updating ω is proposed in [10].

Generally, we expect particles to have strong global search ability in the early evolutionary search while strong local search ability in the late evolutionary search. This makes particles find the global optimal solution.

In order to get better search performance, the dynamic adjustment strategy for ω and μ is proposed as follows [10]:

$$\omega_i^k = k_1 h_i^k + k_2 b_i^k + \omega_0 \quad (11)$$

$$h_i^k = |(\max\{F_{id}^k, F_{id}^{k-1}\} - min\{F_{id}^k, F_{id}^{k-1}\})/f_1| \quad (12)$$

$$b_i^k = 1/n * \sum_{i=1}^{n}(F_i^k - F^{avg})/f_2 \quad (13)$$

$$\mu_i^k = \begin{cases} \left(v_{\max}/v_i^k\right) e^{-\left(k/k_{\max}\right)^2} & if \quad v_i^k > v_{\max} \\ 1 & if \quad v_{\min} < v_i^k < v_{\max} \\ \left(v_{\min}/v_i^k\right) e^{-\left(k/k_{\max}\right)^2} & if \quad v_i^k < v_{\min} \end{cases} \quad (14)$$

where $\omega_0 \in (0,1]$ is the inertia factor which manipulates the impact of the previous velocity history on the current velocity (in most cases is set to 1). In Equation (11), coefficients k_1 and k_2 are typically selected experimentally within the range of [0,1]. In Equation (14), the parameter μ adaptively adjust the value of v^{k+1} by considering the value of v^k.

h_i^k is the speed of evolution,

b_i^k is the average fitness variance of the particle swarms,

F_{id}^k is the fitness value of $pbest_{id}^k$ namely $F(pbest_{id}^k)$,

F_{id}^{k-1} is the fitness value of $pbest_{id}^{k-1}$ namely $F(pbest_{id}^{k-1})$,

f_1 is the normalization function, f_1= max { ΔF_1, ΔF_2,..., ΔF_n}, $\Delta F_i = |F_{id}^k - F_{id}^{k-1}|$,

n is the size of the particle swarms,

F_i^k is the current fitness of the i-th particle,

F^{avg} is the mean fitness of all particles in the swarm at the k-th iteration,

f_2 is the normalization function, f_2=max{$|F_1^k - F^{avg}|$,$|F_2^k - F^{avg}|$,...,$|F_n^k - F^{avg}|$}.

The dynamic adjustment helps PSO not only to avoid the local optimums, but also to enhance the population diversity, which in turn improves the quality of solutions.

In order to compute the RBFNN centers using the improved PSO algorithms such as the method proposed in [9], suppose that a single particle represents a set of k cluster centroid vectors $X = (M_1, M_2, ..., M_k)$, where $M_j = (s_{j1}, ..., s_{jl}, ..., s_{jf})$ refers to the j-th cluster centroid vector of a particle. The M_j has f columns representing the number of features for each pattern of the dataset. Each swarm contains a number of data clustering solutions. The Euclidean distance between each feature of the input pattern and the corresponding cluster centroids is measured by:

$$d(M_{jl}, P_{rl}) = \sqrt{\sum_{l=1}^{f}(S_{jl} - t_{rl})^2} \quad for \ 1 \le j \le k, 1 \le r \le n, 1 \le l \le f \ . \quad (15)$$

After computing all distances for each particle, feature *l* of the pattern *r* is compared with the corresponding feature of the cluster *j*, and then assigns 1 to $Z_{jrl}=1$ when the Euclidean distance for each feature *l* of the pattern *r* is minimum:

$$Z_{jrl}=\begin{cases}1 & d(M_{jl},P_{rl})\ is\ \min\\ 0 & eslewhere\end{cases} \tag{16}$$

In a next step, the mean of the data N_{jl} is computed for each particle according to:

$$N_{jl}=\frac{\sum_{r=1}^{n} t_{rl}\times Z_{jrl}}{\sum_{r=1}^{n} Z_{jrl}} \quad for\ \ 1\le j\le k, 1\le l\le f \tag{17}$$

Moreover, for each feature *l* of the cluster *j*, the Euclidean distances between mean of data N_{jl} and the centroid S_{jl} are computed by:

$$d(N_{jl},S_{jl})=\sqrt{(N_{jl}-S_{jl})^2} \quad for 1\le j\le k, 1\le l\le f \tag{18}$$

Now, the fitness function for each cluster is obtained by summing the calculated distances as follows:

$$F(M_j)=\sum_{l=1}^{f} d(N_{jl},S_{jl}) \quad for 1\le j\le k \tag{19}$$

The proposed method in [10] has been shown in Algorithm 1. Using this algorithm, one can adjust RBF unit centers with *gbest* by iterating for a $k_{\max}$ number of iterations.

Algorithm 1. The pseudocode of the proposed PSO clustering for RBF unit center

For each Particle [*i*] **Do**

Initialize Position vector X[*i*] in the range of maximum and minimum of dataset patterns

Initialize Velocity vector V[*i*] in the range of [-*a*,*a*] (*a* = max(data)-min(data))

Put initial Particle[*i*] into $pbest_{id}[i]$

Next *i*

While maximum iteration is not attained **Do**

For each Particle[*i*] **Do**

For each Cluster[*j*] **Do**

Compute the Fitness Function using Equations (17), (18) and (19)

Next *j*

Next *i*

if run number is greater than 1 **Then**

For each Particle[*i*] **Do**

For each Cluster[*j*] **Do**

if Fitness Function of Particle[*i*]'S Cluster[*j*] is better than Fitness Function of

$pbest_{id}[i]$ 'S Cluster [*j*] **Then**

Put Cluster[*j*] of Particle[*i*] into Cluster[*j*] of $pbest_{id}[i]$

Endif

Next *j*

Next *i*

Endif

For each Cluster[*j*] **Do**

For each Particle[*i*] **Do**

Put the best of $pbests$ in terms of Fitness Function into $gbest$

Next *i*

Next *j*

Compute inertia weight ω using Equation (11)

Compute $\boldsymbol{\mu}$ using Equation (14)

For each Particle [*i*] **Do**

Update Velocity vector V[*i*] using Equation (9)

Update Position vector X[*i*] using Equation (10)

Next *i*

Endwhile

Having processes algorithm, RBF unit centers are adjusted with $gbest$

Moreover, Montazer and Giveki discuss the effect of width adjustment in the performance of the RBFNN in both classification and function approximation problems. Consequently, they proposed a new width adjustment method. Being aware of the high importance of the spatial distribution of the training dataset

and the nonlinearity of the function, whose approximation is desired, they took into account them for the classification problem. Thus, the Euclidean distances between center nodes and the second derivative of the approximated function is used to measure these two factors. Since the width of the center nodes within highly nonlinear areas should be smaller than those of the center nodes in flat areas, Montazer and Giveki proposed to compute the widths of the RBFNN according to Algorithm 2.

Algorithm 2. The Algorithm of the proposed width adjustment

Step 1. Compute the centers of the radial basis functions using our improved PSO clustering

Step 2. Compute mean of squared distances between the centre of cluster *j* and *p-nearest* neighbors

Step 3. Compute coefficient factor: $\boldsymbol{coeff} = \frac{\boldsymbol{d_{max}}}{\sqrt{\boldsymbol{N}}}$,where N is the number of hidden units and $\boldsymbol{d_{max}}$ is the maximum distance between those centers.

Step 4. Find the maximum distance from each center and normalize the distance vector

Step 5. Multiply the distance vector obtained from step 4 by the coefficient factor

Step6. Sum the vector obtained from step 5 with the vector obtained from step 2 as the widths of the improved PSO-OSD RBFNN**.**

In the case of having a function approximation problem, the widths can be computed using Equation (20) as follows:

$$\sigma_i = \frac{d_{\max}}{\sqrt{N}} \cdot \frac{r_i}{\bar{r}} \cdot \left[\frac{1}{1+|f''(c_i)|} \right]^{\frac{1}{4}} \tag{20}$$

For center node c_i, the average distance between this node and its *p*-nearest neighbor nodes is used to measure the spatial distribution feature at this center node, which is defined as following:

$$r_i = \frac{1}{p} \left(\sum_{j=1}^{p} \| c_j - c_i \|^2 \right)^{\frac{1}{2}} \tag{21}$$

where c_j are the *p*-nearest neighboring nodes of c_i. The r_i is the reference dense distance at c_i and the $\bar{r}$ is the average of reference dense distances of all the center nodes and it is computed as follows:

$$\bar{r} = \frac{1}{N} \left(\sum_{i=1}^{N} r_i \right) \tag{22}$$

$f''(c_i)$ is the second derivative of function f and in point c_i and can be computed using the central finite difference method. As the second derivative is used to measure the curvature of the function, the absolute value of the second derivative is used to compare the nonlinearity of different regions in the dataset.

The proposed radial basis function was employed to solve image retrieval problems on large datasets. A more powerful version of the particle swarm optimizer was later proposed in [11]. The proposed PSO was used for adjusting the center of the Gaussian functions in the hidden layer of the RBFNN. The improved RBFNN was employed for solving image classification problem.

The other way of center adjustment in RBFNN that includes a significant portion of these methodologies uses a constructive approach, building the hidden layer incrementally until a criterion is met. Within this context, the application of the orthogonal least squares algorithm has been thoroughly explored [51–54].

3.2.3 Growing RBFNN using orthogonal least squares (OLS)

One of the fine tuned approaches to selecting a subset of data points as the basis function centers is based on the technique of orthogonal least squares. OLS is a forward stepwise regression procedure, where OLS sequentially selects the center that results in the largest reduction of sum-of-square-error at the output. OLS constructs a set of orthogonal vectors Q for the space spanned by the candidate centers. In this orthogonal subspace, computation of pseudo-inverse is avoided since $Q'Q$ becomes diagonal. OLS construct a set of orthogonal vectors Q for the space spanned by basis vectors ϕ_k such that $\Phi = QA$ where, A is an upper triangular matrix. Using this orthogonal representation, the RBF solution is expressed as:

$$T = \Phi W = QG \tag{23}$$

and the LS solution for the weight vector G in the orthogonal space is given as:

$$G = (Q'Q)^{-1}Q'T \tag{24}$$

The above discussions are restricted with center and width selection mechanism except the one in respect to Gaussian kernel. However, there are abundant of proposals developed with their own merits and demerits such as constructive decay [42], resource allocating networks [43], and the minimum description length principle [44]. Recently, Alexandridis *et al.*, [12], introduced a nonsymmetric approach for partitioning the input space. Their experimental outcomes have shown that the nonsymmetric partition can lead to the development of more accurate RBF models, with a smaller number of hidden layer nodes. More elaborate methods have been suggested [45-48] for optimizing the RBF widths in order to improve approximation accuracy.

Taking advantage of the linear connection between the hidden and output layer, most training algorithms calculate the synaptic weights of RBF networks by applying linear regression of the output of the hidden units on the target values. Alternative approaches for calculating the weights include gradient descent methods [6], fuzzy logic [49], and the expectation-maximization algorithm [50].

A few algorithms aspiring to determine all the RBF training parameters in one step have also been proposed in the literature. In [51], a hierarchical Bayesian model is introduced for training RBFs. The model treats all the training parameters as unknown random variables and Bayesian calculation is performed through a reversible jump Markov chain Monte Carlo method, whereas the networks are optimized using a simulated annealing algorithm. In [52], RBF parameters are determined in a one-step algorithm in interpolation problems with equally spaced nodes, after replacing the Euclidean norm associated to Gaussian RBF with a Mahalanobis norm. In [53], all the RBF network parameters, including input weights on the connections between input and hidden layers, are adjusted by a second-order update rule.

Computing optimal values for all the RBF parameters is a rather cumbersome task. Viewing the RBF network training procedure as an optimization problem, one realizes that the objective function usually presents some rather unwelcome properties including, multimodality, non-differentiability and high levels of noise. As these characteristics make use of standard optimization methods inefficient, it is no surprise that a signicant number of studies have focused on optimizing the RBF training procedure through the use of alternative approaches, such as evolutionary-based computation techniques [54]. The resulting methodologies include a genetic algorithm for optimizing the number and coordinates of RBF centers [55], a hybrid multi-logistic methodology applying evolutionary programming for producing RBFs with simpler structures [42], a multi-objective evolutionary algorithm to optimize RBF networks including some new genetic operators in the

evolutionary process [56], and an evolutionary algorithm that performs feature and model selection simultaneously for RBF classiers in reduced computational times [57]. Similarly, PSO is a powerful stochastic optimization algorithm that has been used successfully in conjunction with other computational intelligence tools [58-59]. A PSO-aided orthogonal forward regression algorithm based on leave-one-out criteria is developed in [60] to construct parsimonious RBF networks with tunable nodes. A recursive orthogonal least squares algorithm has been combined with PSO in a novel heuristic structure optimization method for RBF probabilistic networks [61].

In most practical applications, especially in medical diagnosis, the complete training data describing the input-output relationship may not available *a priori*. For these problems, classical batch-learning algorithms are rather infeasible and instead sequential learning is employed. In a sequential learning framework, the training samples arrive one-by-one and the samples are discarded after the learning process. Hence, it requires less memory and computational time for the learning process. In addition, sequential learning algorithms automatically determine the minimal architecture that can accurately approximate the true decision function described by stream of the training samples [33]. Radial basis function (RBF) networks have been extensively used in a sequential learning framework because of their universal approximation ability and simplicity of architecture [62-64]. Recently, there has been renewed interest in single hidden-layered RBF networks with least-square error training criterion, partly due to their modeling ability and partly due to the existence of efficient learning algorithms such as extreme learning machine (ELM) [65], and second-order training methods [6,53]. In recent years, researchers have been focusing on sequential learning algorithms for RBF networks through streams of data.

3.2.4 Training algorithms

Artificial neural networks (ANNs) are considered as universal approximators and applied to solve various problems in different fields, such as image classification [10-11], helicopter sound identification [7-9], and medical diagnosis [12] like diagnosis of heart disease [13], mammograms classification [14], and Parkinson's disease diagnosis [15] . There are two long time discussed issues in ANNs.

1. How large should the network architecture should be?

2. How is a proper training algorithm created or selected?

Issue 1) coincides with a common mistake in neural networks. It is quite easy to obtain error convergence when training larger than required networks, but these networks in most of the cases will respond very poorly to patterns not used for training. Therefore, to avoid overfitting problems, the networks should be as compact as possible [16]. Also, it is often the case that first-order gradient methods are not able to find solutions for these compact networks. In these situations, second-order algorithms are superior [17].

For issue 2), there is no doubt that gradient-based methods are the most straightforward ways to train neural networks. First-order gradient methods [18] are very stable if the learning constant is small, but a tradeoff for good stability is long training time, especially for very accurate approximation. Usually, second-order algorithms show faster convergence and more powerful search ability than first-order algorithms [6,19-21].

For radial basis function (RBF) networks, issue 1) becomes more complex than neural networks. This is because results depend not only on network weights, but also on the locations and widths of RBF units as discussed before [11].

Regarding issue 2), there are a wide variety of learning strategies that have been proposed in literatures for computing the synaptic weights of an RBF network, including :

1) The pseudoinvers (minimum-norm) method [1].

2) The least-mean-square (LMS) method [22].

3) The steepest descent (SD) method [23].

4) The quick propagation (QP) method [6].

The pseudoinvers method is the most direct learning algorithm used in RBF neural networks. This method finds solutions in which results the minimum-norm weight for a given data. A variation of this method, which could be considered as regularization, is that of Poggio and Girosi [24]. The only disadvantage of this method is the need of inversing an N×N matrix, where N is the number of nodes in the RBF network that is often a large number. Thus, this method has high computational and memory usage costs. The LMS method is another method that does not need such vast memory. The most noticeable disadvantage of the LMS algorithm is that using this learning method, the network convergence is not promising. And if it goes to convergence, this process would be very slow. The SD method uses the gradient of error function for each stage of learning process to produce weights of the next stage. The disadvantage of this method is also the slow rate of convergence. The QP method is a way to optimize the SD method. It uses the gradients of the error function in learning process: the current gradient, and the former gradients (quantum). This method excels the learning procedure, although it is still slow.

Substantial amounts of research have been done in the literature to increase the functionality of these base methods. For instance in [6], a set of modified steepest descent methods is presented. Optimum steepest descent (OSD) uses an optimum learning rate in each iteration of the training process [6] as follows:

Let us consider the following definitions:

$$Y_d = [y_{di}], \quad i = 1, \ldots, M \tag{25}$$

where Y_d is the data sample vector and *M* is the number of samples.

$$W = [w_j], \quad j = 1, \ldots, N_h \tag{26}$$

where *W* is the weight vector and N_h is the number of hidden neurons.

$$\Phi = [\phi_j(x_i)], \quad i = 1,2, \ldots, M, \; j = 1,2, \ldots, N_h \tag{27}$$

where ϕ is the general RBF value matrix, which for the Gaussian RBFs we have

$$\phi_j(x_i) = e^{-(x_i - c_j)^2/\sigma_{ij}^2} \tag{28}$$

In a RBF neural network we have :

$$Y = [y_i] = W\phi^T, \quad i = 1, \ldots, M \tag{29}$$

where *Y* is the estimated output vector. It is obvious that the error vector is

$$E = Y_d - Y = Y_d - W\phi^T, \tag{30}$$

and the sum squared error, which should be minimized through the learning process, will be

$$J = \frac{1}{2} EE^T \tag{31}$$

In the conventional SD method, the new weights are computed using the gradient of *J* in the *W* space:

$$OJ = \frac{\partial J}{\partial W} = \frac{\partial((1/2)\mathrm{EE}^T)}{\partial W} = (\mathrm{Y}_d - \mathrm{Y})\frac{\partial Y}{\partial W} = E\frac{\partial(\mathrm{W}\phi^T)}{\partial W} = E\phi \tag{32}$$

$$\Delta W = OJ = E\phi \tag{33}$$

$$W_{new} = W_{old} + \lambda \Delta W, \tag{34}$$

where the coefficient λ is called learning rate (LR), and remains constant through the learning process. It is clear that although the Equation (33) shows the optimum direction of delta weight vector, in the sense of first order estimation, but it still does not specify the optimum length of this vector; and therefore, the optimum learning rate (OLR). To achieve the OLR, the sum-squared error of the new weights should be computed employing Equations (30), (31) and (34):

$$J(W + \lambda \Delta W) = \frac{1}{2}[\mathrm{Y}_d - (W + \lambda \Delta W)\phi^T][\mathrm{Y}_d - (W + \lambda \Delta W)\phi^T]^T = \frac{1}{2}(\mathrm{E} - \lambda \Delta W \phi^T)(\mathrm{E} - \lambda \Delta W \phi^T)^T$$
$$= \frac{1}{2} EE^T - \lambda E\phi \Delta W^T + \frac{1}{2}\lambda^2 \Delta W \phi^T \phi \Delta W^T = A + B\lambda + C\lambda^2, \tag{35}$$

where $A = \frac{1}{2} EE^T$, $B = -E\phi \Delta W^T$ and $C = \frac{1}{2}\Delta W \phi^T \phi \Delta W^T$

are scalar constants. Thus, $J(W + \lambda \Delta W)$ is a quadratic function of λ with coefficients A, B and C. Now, considering these coefficients in detail

$$A = \frac{1}{2} EE^T = \frac{1}{2}\sum_{i=1}^{M} E_i^2 > 0,$$
$$B = -E\phi \Delta W^T \xrightarrow{Eq.(14)} B = -E\phi\phi^T E^T = -(E\phi)(E\phi)^T \le 0$$
$$C = \frac{1}{2} E\phi\phi^T\phi\phi^T E^T \xrightarrow{C=\frac{1}{2}\Delta W\phi^T\phi\Delta W^T} C = \frac{1}{2}(E\phi\phi^T)(E\phi\phi^T)^T \ge 0, \tag{36}$$

$J(\lambda)$ will define a quadratic function of ϕ with positive coefficients of second order term. Thus, it would have a minimum which can be found computing the derivative of $J(\lambda)$:

$$\frac{\partial J}{\partial \lambda} = \frac{\partial(\mathrm{A} + \mathrm{B}\lambda + \mathrm{C}\lambda^2)}{\partial \lambda} = B + 2\lambda C = 0 \xrightarrow{hence} \lambda_{\min} = -\frac{B}{2C} = \frac{(\mathrm{E}\phi)(\mathrm{E}\phi)^T}{(\mathrm{E}\phi\phi^T)(\mathrm{E}\phi\phi^T)^T} \tag{37}$$

This LR minimizes the $J(\lambda)$, and so we can call it the OLR:

$$\lambda_{opt} = \frac{(\mathrm{E}\phi)(\mathrm{E}\phi)^T}{(\mathrm{E}\phi\phi^T)(\mathrm{E}\phi\phi^T)^T} \ge 0 \tag{38}$$

Now the optimum delta weight vector (ODWV) can be determined as

$$\Delta W_{opt} = \lambda_{opt}\Delta W = \frac{(E\phi)(E\phi)^T(E\phi)}{(E\phi\phi^T)(E\phi\phi^T)^T} \quad (39)$$

hence

$$W_{new} = W_{old} + \frac{(E\phi)(E\phi)^T(E\phi)}{(E\phi\phi^T)(E\phi\phi^T)^T} \quad (40)$$

which the initial value for W is chosen randomly.

4. Some applications of RBFNN in medical diagnosis

Classification and prediction are two methods of data analysis that can be used to extract models describing classes or to predict future trends of underlying data. Such analysis can help provide us with a better understanding of the data at large. Mainly, classification predicts categorical (discrete, unordered) labels (that uses training data to generate a single complex rule or mathematical equation that assigns data items to one of several distinct categories), *prediction* models continuous-valued functions. For example, we can build a classification model to categorize bank loan applications as either safe or risky, a patient has cancer (yes, no) based on various medical data, whereas a prediction model to predict the expenditures in dollars of potential customers on computer equipment given their income and occupation. Additionally, classification and prediction have numerous applications, including fraud detection, target marketing, performance prediction, manufacturing, and medical diagnosis. Many classification and prediction methods [33] have been proposed by researchers in neural networks community. However, RBFNNs has attracted a lot of attention in last couple of years. Some of the proposals in this direction are discussed below.

Venkatesan and Anitha [66] have compared RBF neural network with multilayer perceptron network for diabetes diagnosis. Their results show that RBF network performs better than other models. Subashini *et al.* [67] have compared the use of polynomial kernel of SVM and RBFNNs in ascertaining the diagnostic accuracy of cytological data obtained from the Wisconsin breast cancer database. Their research demonstrates that RBFNNs outperformed the polynomial kernel of SVM for correctly classifying the tumors. Chu [68] has applied a novel RBFNNs for cancer classification. He has taken three data sets and the results shows that RBFN is able to achieve 100% accuracy with much fewer genes. Kayaer and Yıldırım [69] have tested the performance of different neural networks for diabetes diagnosis. They concluded that general regression neural network (GRNN) performed better than MLP an RBFNN on Pima Indian Diabetes database. Temurtas *et al*. [70] did a comparative study on pima-diabetes disease diagnosis. For this purpose, a multilayer neural network structure which was trained by Levenberg–Marquardt (LM) algorithm and a radial basis function neural network structure were used. The results of the study were compared with the results of the previous studies reported focusing on diabetes disease diagnosis and using the same UCI machine learning database. As the conclusion, the following results can be summarized:

1. It was seen that neural network structures could be successfully used to help diagnosis of pima-diabetes disease.

2. The classification accuracy of multilayer neural networks with LM obtained by that study using correct training was better than those obtained by other studies for the conventional validation method.

3. For the 10-fold cross-validation method, the classification accuracy of multilayer neural networks with LM obtained by that study using correct training was a bit better than those obtained by other.

4. The results obtained using radial basis function structures are also quite good for Pima Indian diabetes disease diagnostic problem in comparison with the results obtained by the other studies especially for 10-fold cross-validation method.

Breast cancer is the second largest cause of cancer deaths among women. Kiyan and Yildirim [71] have compared the performance of the statistical neural network structures, radial basis network (RBF), general regression neural network (GRNN) and probabilistic neural network (PNN) on the Wisconsin breast cancer data (WBCD). This is a well-used database in machine learning, neural network and signal processing. They observed that RBF and PNN are the best classifiers in training set, however the most important result must be considered with test data since it shows the future performance of the network. GRNN gives the best classification accuracy when the test set is considered. According to overall results, it is seen that the most suitable neural network model for classifying WBCD data is GRNN. Qasem and Shamsuddin [72] have proposed an adaptive evolutionary radial basis function (RBF) network algorithm to evolve accuracy and connections (centers and weights) of RBF networks simultaneously. The problem of hybrid learning of RBF network has been discussed with the multi-objective optimization methods to improve classification accuracy for medical disease diagnosis. In that paper, authors introduce a time variant multi-objective particle swarm optimization (TVMOPSO) of radial basis function (RBF) network for diagnosing the medical diseases (Breast Cancer, Diabetes and Hepatitis). That study applied RBF network training to determine whether RBF networks can be developed using TVMOPSO, and the performance is validated based on accuracy and complexity. Our approach is tested on three standard data sets from UCI machine learning repository. The results show that the proposed approach is a viable alternative and provides an effective means to solve multi-objective RBF network for medical disease diagnosis. It is better than RBF network based on Multi-objective particle swarm optimization and NSGA-II, and also competitive with other methods in the literature. Bascil and Oztekin [73] have used RBFNN classifier for Hepatitis diagnosis. Wu *et al*. [74] have investigated the possibility of using a radial basis function neural network (RBFNN) to accurately recognize and predict the onset of Parkinson's disease tremors in human subjects. The data for training the RBFNN were obtained by means of deep brain electrodes implanted in a Parkinson disease patient's brain. *Wu et al*. concluded that a RBFNN can effectively be used for Parkinson disease diagnosis. Horng *et al*. [75] have introduced a new radial basis function neural networks using firefly optimization algorithm. They evaluated the functionality of their proposed classifier on some medical datasets from the UCI repository. The training procedure involves selecting the optimal values of parameters that are the weights between layer and the output layer, the widths parameters, the center vectors of the radial functions of hidden nodes; and the bias parameters of the neurons of the output layer. The other four algorithms that are gradient descent (GD), genetic algorithm (GA), particle swarm optimization (PSO) and artificial bee colony algorithms have also been implemented for comparisons. The experimental results show that the usage of the firefly algorithm can obtain the satisfactory results over the GD and GA algorithm, but it is not apparent superiority to the PSO and ABC methods form exploring the experimental results of the classifications of UCI datasets.

5. RBFNN simulators

In this Section, we introduce some state-of-the-art tools for implementing RBFNNs. Except MATLAB, all other tools discussed here are open source.

KEEL: Knowledge Extraction based on Evolutionary Learning (KEEL) is an open source (GPLv3) Java software tool which empowers the user to assess the behavior of evolutionary learning and soft computing based techniques for different kinds of DM problems: regression, classification, clustering, pattern mining, and so on [76].

WEKA: In this open source software a normalized Gaussian radial basis function network has been implemented in Java. It uses the k-means clustering algorithm to provide the basis functions and learns either a logistic regression (discrete class problems) or linear regression (numeric class problems). Symmetric multivariate Gaussians are fit to the data from each cluster. If the class is nominal it uses the given number of clusters per class. It standardizes all numeric attributes to zero mean and unit variance.

MATLAB: Two functions have been implemented in MATLAB for radial basis networks. The function *newrb* adds neurons to the hidden layer of a radial basis network until it meets the specified mean squared error goal. The syntax and meaning each argument has been described below.

net = newrb(P,T,goal,spread,MN,DF);

The larger *spread* is, the smoother the function approximation. Too large a spread means a lot of neurons are required to ft a fast-changing function. Too small a spread means many neurons are required to fit a smooth function, and the network might not generalize well. Call *newrb* with different spreads to find the best value for a given problem. Similarly *newrbe* function of MATLAB very quickly designs a radial basis network with zero error on the design vectors. The syntax and meaning of arguments are described below.

net = newrbe(P,T,spread);

The larger the *spread* is, the smoother the function approximation will be. Too large a spread can cause numerical problems.

DTREG: Software for Predictive Modeling and Forecasting implements the most powerful predictive modeling methods that have been developed including, TreeBoost and Decision Tree Forests as well as Neural Networks, Support Vector Machine, Gene Expression Programming and Symbolic Regression, K-Means Clustering, Linear Discriminant Analysis, Linear Regression models and Logistic Regression models. Benchmarks have shown these methods to be highly effective for analyzing and modeling many types of data.

NeuralMachine: NeuralMachine is a general purpose neural network modeling tool with the executable code generator. There are two versions of NeuralMachine - for MS-Windows and a Web-based version (runs in Internet Explorer across the Web). NeuralMachine allows creation of artificial neural networks (ANNs) with one hidden layer. Two types of networks, namely Multilayer perceptron and Radial Basis Function networks are supported in this version. In the case of Radial Basis Network the mapping function is of Gaussian type for the hidden layer and linear for the output layer.

NeuroXL: The NeuroXL software is easy-to-use and intuitive, does not require any prior knowledge of neural networks, and is integrated seamlessly with Microsoft Excel. NeuroXL brings increased precision and accuracy to a wide variety of tasks, including: cluster analysis, stock price prediction, sales forecasting, sports prediction, and much more.

Netlab: The Netlab toolbox is designed to provide the central tools necessary for the simulation of theoretically well founded neural network algorithms and related models for use in teaching, research, and applications development. It consists of a toolbox of MATLAB functions and scripts based on the approach and techniques described in [77], but also including more recent developments in the eld. The Netlab library includes software implementations of a wide range of data analysis techniques, many of which are not yet available in standard neural network simulation packages. Netlab works with MATLAB version 5.0 and higher but only needs core MATLAB (i.e., no other toolboxes are required). It is not compatible with earlier versions of MATLAB.

6. Conclusion and future research

This paper investigates a review on various methods proposed for designing RBFNNs. Besides, we pointed to some applications of this popular and powerful classifier in medical disease diagnosis. At first, the architecture of RBFNNs was explained, where the central problems are based on the appropriate centers and widths adjustments for Gaussian functions, determination of the number of hidden neurons, and the optimization of weights associated between hidden and output layer neurons. As far as the adjustment of the centers and the widths are concerned, k-means clustering, methahurestic based clustering (such as PSO-

Clustering) and *p*-nearest neighbor algorithm have been widely used for solving these problems. Similarly, there are lots of independent proposals have been discussed for solving the problem of selection of hidden layer neurons. Since RBFNN expansions are linearly dependents on the weights, therefore, a globally optimum least squares interpolation of non-linear maps can be achieved. However, singular value decomposition (SVD) is widely used to optimize the weights.

Secondly, we have discussed various training algorithms of RBFNNs in classification and prediction that have been used for solving wide variety of problems. Additionally some domain specific usage of RBFNNs is presented.

Thirdly, we have introduced some open source software tools for testing RBFNNs. To the best of our knowledge, there is no such report/document has been reported for a fair comparison of tools in connection to the successful application of RBFNNs.

RBFNNs still open avenues for new kernels along many issues still remain unsolved. In addition RBFNNs need to be systematically evaluated and compared with other new and traditional tools. We believe that the multidisciplinary nature of RBFNNs will generate more research activities and bring about more fruitful outcomes in the future. For instance, there are many new and powerful metaheuristic optimization algorithms such as bat optimization that can be used for determining centers, widths and weights of the RBFNN.

References

1. Broom head, D. S., & Lowe, D. (1988). *Radial basis functions, multi-variable functional interpolation and adaptive networks* (No. RSRE-MEMO-4148). ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (UNITED KINGDOM).

2. Yu, H., Xie, T., Paszczynski, S., & Wilamowski, B. M. (2011). Advantages of radial basis function networks for dynamic system design. *IEEE Transactions on Industrial Electronics*, *58*(12), 5438-5450.

3. Tan, K. K., & Tang, K. Z. (2005). Adaptive online correction and interpolation of quadrature encoder signals using radial basis functions. *IEEE Transactions on Control Systems Technology*, *13*(3), 370-377.

4. Mu, T., & Nandi, A. K. (2005). Detection of breast cancer using v-SVM and RBF networks with self organized selection of centers.

5. Okamato, K., Ozawa, S., & Abe, S. (2003, July). A fast incremental learning algorithm of RBF networks with long-term memory. In *Neural Networks, 2003. Proceedings of the International Joint Conference on* (Vol. 1, pp. 102-107).

6. Montazer, G. A., Sabzevari, R., & Khatir, H. G. (2007). Improvement of learning algorithms for RBF neural networks in a helicopter sound identification system. *Neurocomputing*, *71*(1), 167-173.

7. Montazer, G. A., Sabzevari, R., & Ghorbani, F. (2009). Three-phase strategy for the OSD learning method in RBF neural networks. *Neurocomputing*, *72*(7), 1797-1802.

8. Montazer, G. A., Khoshniat, H., & Fathi, V. (2013). Improvement of RBF neural networks using Fuzzy-OSD algorithm in an online radar pulse classification system. *Applied Soft Computing*, *13*(9), 3831-3838.

9. Fathi, V., & Montazer, G. A. (2013). An improvement in RBF learning algorithm based on PSO for real time applications. *Neurocomputing*, *111*, 169-176.

10. Montazer, G. A., & Giveki, D. (2015). An improved radial basis function neural network for object image retrieval. *Neurocomputing*, *168*, 221-233.

11. Montazer, G. A., & Giveki, D. (2017). Scene Classification Using Multi-Resolution WAHOLB Features and Neural Network Classifier. *Neural Processing Letters*, 1-24.

12. Alexandridis, A., & Chondrodima, E. (2014). A medical diagnostic tool based on radial basis function classifiers and evolutionary simulated annealing. *Journal of biomedical informatics*, *49*, 61-72.

13. Alsalamah, M., Amin, S., & Halloran, J. (2014, December). Diagnosis of heart disease by using a radial basis function network classification technique on patients' medical records. In *RF and Wireless Technologies for Biomedical and Healthcare Applications (IMWS-Bio), 2014 IEEE MTT-S International Microwave Workshop Series on* (pp. 1-4).

14. Pratiwi, M., Harefa, J., & Nanda, S. (2015). Mammograms classification using gray-level co-occurrence matrix and radial basis function neural network. *Procedia Computer Science*, *59*, 83-91.

15. Er, O., Cetin, O., Bascil, M. S., & Temurtas, F. (2016). A Comparative Study on Parkinson's Disease Diagnosis Using Neural Networks and Artificial Immune System. *Journal of Medical Imaging and Health Informatics*, *6*(1), 264-268.

16. Wilamowski, B. M. (2009). Neural network architectures and learning algorithms. *IEEE Industrial Electronics Magazine*, *3*(4).

17. Fnaiech, N., Fnaiech, F., Jervis, B. W., & Cheriet, M. (2009). The combined statistical stepwise and iterative neural network pruning algorithm. *Intelligent Automation & Soft Computing*, *15*(4), 573-589.

18. Fahlman, S. E. (1988). An empirical study of learning speed in back-propagation networks.

19. Hagan, M. T., & Menhaj, M. B. (1994). Training feedforward networks with the Marquardt algorithm. *IEEE transactions on Neural Networks*, *5*(6), 989-993.

20. Wilamowski, B. M., & Yu, H. (2010). Neural network learning without back propagation. *IEEE Transactions on Neural Networks*, *21*(11), 1793-1803.

21. Wilamowski, B. M., & Yu, H. (2010). Improved computation for Levenberg–Marquardt training. *IEEE transactions on neural networks*, *21*(6), 930-937.

22. Haykin, S. S., Haykin, S. S., Haykin, S. S., & Haykin, S. S. (2009). *Neural networks and learning machines* (Vol. 3). Upper Saddle River, NJ, USA:: Pearson.

23. Neruda, R., & Kudová, P. (2005). Learning methods for radial basis function networks. *Future Generation Computer Systems*, *21*(7), 1131-1142.

24. Poggio, T., & Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, *78*(9), 1481-1497.

25. Powell M.J.D., Radial Basis Functions for Multivariable Interpolation: A Review. In: J.C. Mason, M.G. Cox (Eds.), Algorithms for the Approximation, Clarendon Press, 1987, 143-167

26. Gyor L., Kohler M., Krzyzak A., Walk H., A Distribution Free Theory of Non-Parametric Regression. Springer Verlag, New York Inc., 2002

27. Broom head D.S., Lowe D., Multi-variable Functional Interpolation and Adaptive Networks. Complex Systems, 2, 321-355, 1988

28. Park J., Sandberg I.W., Universal Approximation Using Radial Basis Function Networks. Neural Computation, 3(2), 246-257, 1991.

29. Franke C., Schaback, R., Convergence Orders of Meshless Collocation Methods using Radial Basis Functions. Advances in Computational Mathematics, 8(4), 381-399, 1998

30. Baxter B.J.C., The Interpolation Theory of Radial Basis Functions. University of Cambridge, Cambridge, 1992.

31. Acosta F.M.A., Radial Basis Functions and Related Models: An Overview. Signal Processing, 45(1), 37-58, 1995.

32. Lim E.A., Zainuddin Z., An Improved Fast Training Algorithm for RBF Networks using Symmetry-Based Fuzzy C-Means Clustering. Matematika, 24(2), 141-148, 2008

33. Dash, C. S. K., Behera, A. K., Dehuri, S., & Cho, S. B. (2016). Radial basis function neural networks: a topical state-of-the-art survey. *Open Comput Sci*, *6*, 33-63.

34. Hunt K.J., Haas R., Murray-Smith R., Extending the Functional Equivalence of Radial Basis Function Networks and Fuzzy Inference Systems. IEEE Trans Neural Network, 7, 776-781, 1996.

35. Gu Z., Li H., Sun Y., Chen Y., Fuzzy Radius Basis Function Neural Network Based Vector Control of Permanent Magnet Synchronous Motor. In Proceedings of IEEE International Conference on Mechatronics and Automation (ICMA 2008), 224-229, 2008.

36. Kong F., Zhang Z., Liu Y., Selection of Logistics Service Provider Based on Fuzzy RBF Neural Networks. In Proceedings of IEEE International Conference on Automation and Logistics (ICAL 2008), 1189-1192, 2008.

37. Li X., Dong J., Zhang Y., Modeling and Aying of RBF Neural Network Based on Fuzzy Clustering and Pseudo-Inverse Method. In Proceedings of International Conference on Information Engineering and Computer Science (ICIECS 2009), 1-4, 2009.

38. Kamalabady A.S., Salahshoor K., New SISO and MISO Adaptive Nonlinear Predictive Controllers Based on Self Organizing RBF Neural Networks. In Proceedings of 3rd International Symposium on Communications, Control, and Signal Processing (ISCCSP 2008), 703-708, 2008.

39. Liu H., He J., The Aication of Dynamic K-means Clustering Algorithm in the Center Selection of RBF Neural Networks. 3rd International Conference on Genetic and Evolutionary Computing, 2009. WGEC'09, 488-491, 2009.

40. Zainuddin Z., Lye W.K., Improving RBF Networks Classification Performance by using K-Harmonic Means. World Academy of Science, Engineering and Technology, 62, 983-986, 2010.

41. Huilan J., Ying G., Dongwei L., Jianqiang X., Self-adaptive Clustering Algorithm Based RBF Neural Network and its Aication in the Fault Diagnosis of Power Systems. In Proceedings of IEEE/PES Transmission and Distribution Conference and Exhibition: Asia and Pacic, Dalian, 1-6, 2005.

42. Fernandez-Navarro F., Hervas-Martinez C., Cruz-Ramirez M., Gutierrez P.A., Valero A., Evolutionary q-Gaussian Radial Basis Function Neural Network to Determine the Microbial Growth/no Growth Interface of Staphylococcus Aureus. Aied Soft Computing, 11(3), 3012-3020, 2011.

43. Yingwei L., Sundararajan N., Saratchandran P., A Sequential Learning Scheme for Function Approximation using Minimal Radial Basis Function Neural Networks. Neural Computation, 9(2),461-478, 1997.

44. Leonardis A., Bischof H., An Efficient MDL-Based Construction of RBF Networks. Neural Networks, 11(5), 963-973, 1998.

45. Orr, M., Optimising the Widths of Radial Basis Functions. In Proc. 5th Brazilian Symp. Neural Netw., 26–29, 1998.

46. Yao W., Chen X., Zhao Y., Tooren M.V., Concurrent subspace width optimization method for RBF neural network modeling, IEEE Trans. Neural Netw. Learn. Syst., 23(2), 247–259, 2012.

47. Yao W., Chen X., Tooren M.V., Wei Y., Euclidean Distance and Second Derivative Based Widths Optimization of Radial Basis Function Neural Networks. In Proc. Int. Joint Conf. Neural Netw., 18-23 July 2010, Barcelona, Spain, 1–8, 2010.

48. Orr M.J., Regularization in the Selection of Radial Basis Function Centers. Neural Computation, 7(3), 606-623, 1995.

49. Zhang D., Deng L.F., Cai K.Y., So A., Fuzzy Nonlinear Regression with Fuzzified Radial Basis Function Network. IEEE Trans. FuzzySyst., 13(6), 742–760, 2005.

50. Langari R., Wang L., Yen J., Radial Basis Function Networks, Regression Weights, and the Expectation-Maximization Algorithm. IEEE Trans. Syst., Man., Cybern. A, Syst. Humans, 27(5), 613–623, 1997.

51. Andrieu C., Freitas N.D., Doucet A., Robust Full Bayesian Learning for Radial Basis Networks. Neural Comput., 13(10), 2359–2407, 2001.

52. Huan H.X., Hien D.T.T., Tue H.H., Efficient Algorithm for Training Interpolation RBF Networks with Equally Spaced Nodes. IEEE Trans. Neural Netw., 22(6), 982–988, 2011.

53. Xie T., Yu H., Hewlett J., Rozycki P., Wilamowski B., Fast and Efficient Second-Order Method for Training Radial Basis Function Networks. IEEE Trans. Neural Netw. Learn. Syst., 23(4), 609–619, 2012.

54. Gutiérrez P.A., Hervás-Martínez C., Martínez-Estudillo F.J., Logistic Regression by Means of Evolutionary Radial Basis Function Neural Networks. IEEE Trans. Neural Netw., 22(2), 246–263, 2011.

55. Sarimveis H., Alexandridis A., Mazarakis S., Bafas G., A New Algorithm for Developing Dynamic Radial Basis Function Neural Network Models Based on Genetic Algorithms. Comput. Chem. Eng., 28(1–2), 209–217, 2004.

56. Gonzalez J., Rojas I., Ortega J., Pomares H., Fernandez F., Javier D., Antonio F., Multi-objective Evolutionary Optimization of the Size, Shape, and Position Parameters of Radial Basis Function Networks for Function Approximation. IEEE Transactions on Neural Networks, 14(6), 1478-1495, 2003.

57. Buchtala O., Klimek M., Sick B., Evolutionary Optimization of Radial Basis Function Classifiers for Data Mining Aications. IEEE Trans. Syst. Man. Cybern. B Cybern., 35(5), 928–947, 2005.

58. Valdez F., Melin P., Castillo O., Evolutionary Method Combining Particle Swarm Optimisation and Genetic Algorithms Using Fuzzy Logic for Parameter Adaptation and Aggregation: The Case Neural Network Optimisation for Face Recognition. Int. J. Artif. Intell. Soft Comput., 2(1–2), 77–102, 2010.

59. Valdez F., Melin P., Castillo O., An Improved Evolutionary Method with Fuzzy Logic for Combining Particle Swarm Optimization and Genetic Algorithms. A. Soft Comput., 11(2), 2625–2632, 2011.

60. Chen S., Hong X., Harris C.J., Particle Swarm Optimization Aided Orthogonal Forward Regression for Unied Data Modeling. IEEETrans. Evol. Comput.,14(4), 477–499, 2010.

61. Huang D.S., Du J.X., A Constructive Hybrid Structure Optimization Methodology for Radial Basis Probabilistic Neural Networks. IEEE Trans. Neural Networks, 19(12), 2099–2115, 2008.

62. Suresh S., Sundararajan N., Saratchandran P., A Sequential Multicategory Classifier Using Radial Basis Function Networks. Neurocomputing, 71(1), 1345–1358, 2008.

63. Huang G.B., Saratchandran P., Sundararajan N., An Ecient Sequential Learning Algorithm for] Growing and Pruning RBF (GAPRBF) Networks. IEEE Transactions on Systems, Man, Cybernetics, 34(6), 2284-2292, 2004.

64. Huang G., Saratchandran P., Sundararajan N., A Generalized Growing and Pruning RBF (GGAP-RBF) Neural Network for Function Aoximation. IEEE Transactions on Neural Networks, 16, 57- 67, 2005.

65. Huang G.-B., Zhou H., Ding X., Zhang R., Extreme learning machine for regression and multiclass classification. IEEE Trans. Syst. Man Cybern. B Cybern., 42(2), 513–529, 2012.

66. Venkatesan, P., & Anitha, S. (2006). Application of a radial basis function neural network for diagnosis of diabetes mellitus. *Current Science*, *91*(9), 1195-1199.

67. Subashini T.S., Ramalingam V., Palanivel S., Breast Mass Classification Based on Cytological Patterns using RBFNN and SVM. Expert Systems with Aications, 36, 5284-5290, 2008.

68. Chu F., Wang L., Aying RBF Neural Networks to Cancer Classification Based on Gene Expressions. In Proceedings of Int. Joint Conference on Neural Networks (IJCNN'06), 1930-1934, 2006.

69. Kayaer, K., & Yıldırım, T. (2003, June). Medical diagnosis on Pima Indian diabetes using general regression neural networks. In *Proceedings of the international conference on artificial neural networks and neural information processing (ICANN/ICONIP)* (pp. 181-184).

70. Temurtas, H., Yumusak, N., & Temurtas, F. (2009). A comparative study on diabetes disease diagnosis using neural networks. *Expert Systems with applications*, *36*(4), 8610-8615.

71. Kiyan, T., & Yildirim, T. (2004). Breast cancer diagnosis using statistical neural networks. *Journal of electrical & electronics engineering*, *4*(2), 1149-1153.

72. Qasem, S. N., & Shamsuddin, S. M. (2011). Radial basis function network based on time variant multi-objective particle swarm optimization for medical diseases diagnosis. *Applied Soft Computing*, *11*(1), 1427-1438.

73. Bascil, M. S., & Oztekin, H. (2012). A study on hepatitis disease diagnosis using probabilistic neural network. *Journal of medical systems*, *36*(3), 1603-1606.

74. Wu, D., Warwick, K., Ma, Z., Burgess, J. G., Pan, S., & Aziz, T. Z. (2010). Prediction of Parkinson's disease tremor onset using radial basis function neural networks. *Expert Systems with Applications*, *37*(4), 2923-2928.

75. Horng, M. H., Lee, Y. X., Lee, M. C., & Liou, R. J. (2012). Firefly metaheuristic algorithm for training the radial basis function network for data classification and disease diagnosis. *Theory and new applications of swarm intelligence*, 115-132.

76. Alcalá-Fdez J., Sánchez L., García S., del Jesus M.J., Ventura S.,Garrell J.M., Otero J., Romero C., Bacardit J., Rivas V.M., Fernández J.C., Herrera F., KEEL: A Software Tool to Assess Evolutionary Algorithms to Data Mining Problems. Soft Computing, 13, 307- 318, 2009.

77. Bishop C., Neural Networks for Pattern Recognition. Oxford University Press, USA, 1995